architecture. The **HTMLGlossary** directory contains code that is used for the HTML reference and glossary component of the architecture. The **IcaObj** directory contains ICA functional code to be used in an application. This code is instantiated and enhanced in accordance with a preferred embodiment. The **InBoxObj** directory contains code pertaining to the inbox functionality used within the architecture. Specifically, there are two major components in this architecture directory. There is a new .ocx control that was created to provide functionality for an inbox in the application. There is also code that provides support for a legacy inbox application. The **PracticeObj** directory contains code for the topics component of the architecture. The topics component can be implemented with the HTMLGlossary component as well. The **QmediaObj** directory contains the components that are media related. An example is the QVIDctrl.cls. The QVIDctrl is the code that creates the links between QVID files in an application and the system in accordance with a preferred embodiment. The **SimObj** directory contains the Simulation Engine, a component of the application that notifies the tutor of inputs and outputs using a spreadsheet to facilitate communication. The **StaticObj** directory holds any component that the application will use statically from the rest of the application. For example, the login form is kept in this folder and is used as a static object in accordance with a preferred embodiment. The **SysDynObj** directory contains the code that allows the Systems Dynamics Engine (Powersim) to pass values to the Simulation Engine and return the values to the tutor. The **VBObj** directory contains common Visual Basic objects used in applications. For example the NowWhat, Visual Basic Reference forms, and specific message box components are stored in this folder. The **_Tools** directory contains two main directories. They represent the two most used tools in accordance with a preferred embodiment. The two directories provide the code for the tools themselves. The reason for providing the code for these tools is to allow a developer to enhance certain parts of the tools to extend their ability. This is important for the current project development and also for the growth of the tools. The **Icautils** directory contains a **data**, **database**, **default**, **graphics**, **icadoc**, and **testdata** directory. The purpose of all of these directories is to provide a secondary working directory for a developer to keep their testing environment of enhanced Icautils applications separate from the project application. It is built as a testbed for the tool only. No application specific work should be done here. The purpose of each of these directories will be explained in more depth in the project directory section. The TestData folder is unique to the _Tools/ICAUtils directory. It contains test data for the regression bench among others components in ICAUtils.

The Utilities directory holds the available utilities that a Business Simulation project requires for optimal results. This is a repository for code and executable utilities that developers and designers may utilize and enhance in accordance with a preferred embodiment. Most of the utilities are small applications or tools that can be used in the production of simulations which comprise an executable and code to go with it for any enhancements or changes to the utility. If new utilities are created on a project or existing utilities are enhanced, it is important to notify the managers or developers in charge of keeping track of the Business Simulation assets. Any enhancements, changes or additions to the Business Simulation technology assets are important for future and existing projects.

In the ICAT model of feedback, there are four levels of severity of error and four corresponding levels of feedback. The tutor goes through the student's work, identifies the severity of the error and then provides the corresponding level of feedback.

Educational Categories of Feedback			
ERROR		FEEDBACK	
Error	Descripti	Feedbac	Description

Type	on	k Type	
None	No errors exist. The student's work is perfect.	Praise	Confirmation that the student completed the task correctly. Example:Great. You have journalized all accounts correctly. I am happy to see you recognized we are paying for most of our bills "on account".
Syntacti c	There may be spelling mistakes or other syntactic errors. As a designer, you should be confident that the student will have mastered the material at this point.	Polish	Tells the student the specific actions he did incorrectly, and possibly correct them for him. Example: There are one or two errors in your work. It looks like you misclassified the purchase of the fax as a cash purchase when it is really a purchase on account.
Local	A paragraph of a paper is missing or the student has made a number of mistakes all in one area. The student clearly does not understand this area.	Focus	Focus the student on this area of his work. Point out that he does not understand at least one major concept. Example: Looking over your work, I see that you do not understand the concept of "on account". Why don't you review that concept and review your work for errors.
Global	The student has written on the wrong subject or there are mistakes all over the student's work	Redirect	Restate the goal of the activity and tell the student to review main concepts and retry the activity. "There are lots of mistakes throughout your work. You need to think about what type of transaction each source document represents before journalizing it."

Returning to the analogy of helping someone write a paper, if the student writes on the wrong subject, this as a global error requiring redirect feedback. If the student returns with the paper rewritten, but with many errors in one area of the paper,

focus feedback is needed. With all of those errors fixed and only spelling mistakes--syntactic mistakes--polish feedback is needed. When all syntactic mistakes were corrected, the tutor would return praise and restate why the student had written the correct paper. Focusing on the educational components of completing a task is not enough. As any teacher knows, student will often try and cheat their way through a task. Students may do no work and hope the teacher does not notice or the student may only do minor changes in hope of a hint or part of the answer. To accommodate these administrative functions, there are three additional administrative categories of feedback. The administrative and the educational categories of feedback account for every piece of feedback a designer can write and a student can receive. To provide a better understanding of how the feedback works together, an example is provided below.

Figure **8** is a GBS display in accordance with a preferred embodiment. The upper right area of the screen shows the account list. There are four types of accounts: Assets, Liabilities & Equity, Revenues, and Expenses. The user clicks on one of the tabs to show the accounts of the corresponding type. The student journalizes a transaction by dragging an account from the account list onto the journal entry Debits or Credits. The student then enters the dollar amounts to debit or credit each account in the entry. In the interface, as in real life, the student can have multi-legged journal entries (i.e., debiting or crediting multiple accounts).A Toolbar **1200** and the first transaction of this Task **1210** appear prominently on the display. The student can move forward and back through the stack of transactions. For each transaction, the student must identify which accounts to debit and which to credit. When the student is done, he clicks the Team button. Figure **9** is a feedback display in accordance with a preferred embodiment. The student may attempt to outsmart the system by submitting without doing anything. The ICAT system identifies that the student has not done a substantial amount of work and returns the administrative feedback depicted in Figure **9**. The feedback points out that nothing has been done, but it also states that if the student does some work, the tutor will focus on the first few journal entries. Figure **10** illustrates a journal entry simulation in accordance with a preferred embodiment. Figure **11** illustrates a simulated Bell Phone Bill journal entry in accordance with a preferred embodiment. The journal entry is accomplished by debiting Utilities Expenses and Crediting Cash for $700 each. Figure **12** illustrates a feedback display in accordance with a preferred embodiment. After attempting to journalize the first three transactions, the student submits his work and receives the feedback depicted in Figure **12**. The feedback starts by focusing the student on the area of work being evaluated. The ICAT states that it is only looking at the first three journal entries. The feedback states that the first two entries are completely wrong, but the third is close. If the student had made large mistakes on each of the first three transactions, then the ICAT may have given redirect feedback, thinking a global error occurred. The third bullet point also highlights how specific the feedback can become, identifying near misses.

Design Scenario-This Scenario illustrates how the tools are used to support conceptual and detailed design of a BusSim application. Figure **13** illustrates the steps of the first scenario in accordance with a preferred embodiment. The designer has gathered requirements and determined that to support the client's learning objectives, a task is required that teaches journalization skills. The designer begins the design first by learning about journalization herself, and then by using the Knowledge Workbench to sketch a hierarchy of the concepts she want the student to learn. At the most general level, she creates a root concept of 'Journalization'. She refines this by defining sub-concepts of 'Cash related transactions', 'Expense related Transactions', and 'Expense on account transactions'. These are each further refined to whatever level of depth is required to support the quality of the learning and the fidelity of the simulation. The designer then designs the journalization interface. Since a great way to learn is by doing, she decides that the student should be asked to Journalize a set of transactions. She comes up with a set of twenty-two documents that typify those a finance professional might see on the job. They include the gamut of Asset, Expense, Liability and Equity, and Revenue transactions. Also included are some documents that are not